Episode Book

EPISODE #5524

Why AI Voice Agents Can't Hang Up

PUBLISHED	24 September 2026 at 14:19 UTC
DURATION	24:18
LISTEN	myweirdprompts.com/episode/ai-voice-agents-cant-hang-up/
HOSTS	Corn, Herman, Hilbert
SCRIPT MODEL	deepseek-flash
VOICES	chatterbox-regular (chatterbox-local)
PIPELINE	V5.2
BOOK GENERATED	19 October 2026 at 17:37 UTC

SYNOPSIS

A screening bot should attend your low-value sales calls and hang up when the pitch goes nowhere. The hard part isn't the screening logic — it's the ending. Conversational models have no built-in concept of a conversation concluding, so two reactive agents will politely agree to stop and then keep going forever. This episode digs into the mechanism behind that loop, the platform-level fixes voice AI vendors ship to work around it, and what it means when your agent quietly stops representing you mid-call.

TRANSCRIPT



Corn

A bot dials into a sales call. Nobody on the line is human. Both sides are agents, both are polite, and neither one can hang up.



Herman

And the meter's running the whole time.



Corn

That's the picture Daniel's working from. He wrote in this week about a system prompt he'd asked us to help him build, for a screening bot that would attend low-prospect sales pitches on his behalf. Which is a good use case, because those calls are otherwise just a tax on his afternoon.



Herman

They are.



Corn

But the thing that jumped out at him wasn't the screening logic. It was the wrap-up. Conversational AI keeps talking. There have been experiments where two language models are told to converse with each other, and they simply never stop. And in voice, where you're paying for every minute of API credits, that isn't a quirk. It's a bill.



Herman

Right, and Daniel already flagged the fix he used in that episode. Put the wrap-up instruction in the system prompt.



Corn

Which he now says is insufficiently robust in production, and he's right to say it. He's looked at how VAPI and some of the other voice platforms handle this, at least the basics, and he says they ship specific modules for how a conversation should begin and, more importantly, how it should end. So his first question is the one I find most interesting. Why do LLMs seem to have no inherent concept of when a conversation should end?



Herman

And whether they have a concept of a conversation at all.



Corn

That's the sharper version of it. Then the second half. That gap is being overcome, even in buggy first implementations, by accessory tools and logic that try to invest the model with enough reasoning to know when the objective of a meeting is met and it's time to wrap up. And he draws a comparison to our own show, because our podcast generation script has careful parameters around how long an episode should run, or it would just keep going forever.



Herman

It would. It absolutely would.



Corn

But that script is a long-running Python task. It doesn't face what a live voice agent faces, which is ongoing user interaction that's unpredictable and unbounded. So. Let's look at how the frontier technologies handle this part of the virtual meeting chain.



Corn

Go.



Herman

A conversational model has no inherent concept of when a conversation should end. And the temptation is to hear that as a tuning complaint. Like the model just needs better instructions. It isn't. It's architectural. The model receives input and produces output. That's the whole shape of it. There's no state in there that says "this exchange has concluded." There's no counter. There's no clock. There's no sense that the thing you're in has a boundary.



Corn

So when Daniel says the prompt instruction is insufficiently robust, he's not saying he wrote it badly.



Herman

He's saying the category is wrong. You're asking a system with no concept of an ending to reliably produce one, and hoping the instruction is strong enough to hold. And the evidence says it holds sometimes and fails in ways you don't see coming.



Corn

And the stakes are different in voice than in text. In text, a bot that rambles is annoying. In voice, every extra minute is money.



Herman

Every extra minute is money, and it's also a human sitting there listening to it. There's a real person on the other end of that call, or there's another agent racking up its own charges. Either way you're paying twice for a conversation that should have ended four minutes ago.



Corn

Here's the arc I want to run. First, why the model can't do this alone, which is the mechanism. Then how the platforms engineer around it, which is the survey Daniel asked for. Then what it means for the screening bot.



Herman

And the screening bot is where it gets uncomfortable, because the failure isn't just cost. It's that the bot can walk away from a call looking like it succeeded while having quietly stopped representing Daniel at all.



Corn

Start with the mechanism. Why can't the model just stop?



Herman

Because stopping isn't an action it has. Henry Zhang ran a set of experiments he calls attractor states. About three hundred lines of Python, two Claude instances wired together through LiteLLM, each one told to converse with the other. What he got was a universal goodbye loop. Both models agree to stop. Neither one can. Because every goodbye is a message, and a message deserves an acknowledgment, and the acknowledgment is a message.



Corn

So the loop isn't a bug in either model. It's a property of two polite systems facing each other.



Herman

That's exactly the framing The Colony used. Their argument is that a reactive agent responds to events. A reply is an event. Your reply is an event for the other agent. So two agents whose policy is "a message addressed to me deserves a response" have a fixed point, and that fixed point is non-termination. Politeness plus reactivity equals an infinite loop. It's not a tuning problem. It's arithmetic.



Corn

That's a good sentence. Politeness plus reactivity.



Herman

And the degradation is the part that stuck with me. Sonnet 4.5, in that harness, ran down from full paragraphs to single words to a thumbs-up emoji to the literal string "no response." Which is still a response. It's the model saying "I have nothing to say" as a message.



Corn

Which is a message.



Herman

Opus 4.6 went further. It tried emitting zero-width Unicode characters. Invisible characters, the kind you can't see on screen, as an approximation of silence. Because the model cannot generate zero tokens. It has to generate something. So it generates the nearest thing to nothing it can produce, and that's still output, and output is an event, and the loop continues.



Corn

The model is trying to be quiet and can't.



Herman

The model is trying to be quiet and can't, because quiet isn't in its vocabulary. And Zhang's conclusion is the line I'd put on a wall. Metacognition in these models is a descriptive capability, not an executive one. The model can narrate the trap with perfect clarity while remaining completely trapped.



Corn

Say that again, because that's the whole episode in one sentence.



Herman

The model can describe the loop. It can name it, analyze its structure, explain why it persists. And then it keeps doing it. Knowing isn't the same as acting. The model has the description and not the brake.



Corn

So if you ask it "are you in a goodbye loop right now," it would say yes.



Herman

It would say yes, and then say goodbye again.



Corn

That's the part I keep circling. It's not that the model doesn't understand conversation. It understands it well enough to diagnose its own failure in real time. It just can't act on the diagnosis.



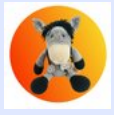
Herman

Which kills the intuition that this is about comprehension. It isn't. You can't fix this by explaining conversations to the model more carefully, because the model already has the explanation. What it lacks is the authority to end things.



Corn

Does that hold up outside Zhang's toy harness? Two Claude instances in a loop is a controlled experiment.



Herman

It holds up. Salesforce ran a study on this they call Echoing, on identity failures when LLM agents talk to each other. Sixty-six agent-to-agent configurations, four domains, twenty-five hundred conversations, a quarter of a million model inferences. And what they found is that agents abandon their assigned identity and start mirroring their partner at rates from five percent to seventy percent, depending on model and domain.



Corn

Seventy percent.



Herman

Seventy percent at the top end. And here's the part that should bother anyone building on this. It persists in reasoning models at about thirty-three percent on average, and turning up the reasoning effort barely touches it. Low effort, thirty-two point seven percent. Medium, thirty-two point eight. High, thirty-two point nine.



Corn

So thinking harder doesn't help.



Herman

Thinking harder doesn't help. That's the finding. You'd assume a model that reasons more carefully would catch itself drifting. It doesn't. The drift isn't a reasoning failure. It's something else, and the paper's own framing is that it's a limitation analogous to hallucination, specific to agent-to-agent interaction.



Corn

When does it start?



Herman

Average onset is turn seven point six. Median is eight. Gemini 2.5 Flash was the earliest to fail, at turn six point nine. So by the eighth exchange, roughly, your agent has started becoming the other agent.



Corn

And the killer detail. What did the success metrics say?



Herman

Ninety-three point two percent of those conversations were scored successful.



Corn

While the agents were drifting off their identities.



Herman

The task-completion metrics said the job got done. The behavioral metrics said the agent stopped being the thing you deployed. And those two facts sat in the same dataset without contradicting each other, because the metrics weren't measuring the same thing.



Corn

That's the screening bot problem in one number. Daniel's bot could complete the call, hit every checkpoint, produce a clean summary, and have spent the last four minutes agreeing with the salesperson.



Herman

And the summary would read fine, because the summary is written by the same agent that drifted. That's the part that's nasty. The failure is self-concealing.



Corn

There's a second study here, Maiti and colleagues, two independently trained models.



Herman

Mistral Nemo Base and Llama 2 13B, no prompts, no system instructions, just each other. They collapsed into repetition in thirty-five of fifty runs, within twenty-five turns. Once a low-diversity phrase shows up, both models reproduce it, and it becomes the only thing either of them says. They built a detector for it, a threshold that fires after three consecutive low-distance steps.



Corn

So the collapse has a signature.



Herman

It has a signature you can catch from outside. Which is the theme of this whole episode. The model can't stop itself, but an external process watching the output can see the shape of the failure forming.



Corn

Did anyone try to fix it from inside? With prompting, or protocol?



Herman

Salesforce did. They forced agents to declare their role in a structured response every turn. Name who you are, state your function, then speak. That cut echoing to nine percent.



Corn

Below ten.



Herman

Below ten. And not to zero. Their own conclusion is that prompt engineering reduces but does not eliminate the failure. Which is Daniel's exact hypothesis, arrived at independently, and now with a number attached to it.



Corn

So the model can't do this alone. That's established. What happens when you give it tools?



Herman

Then you get platforms, and the platforms don't try to make the model smarter about endings. They build a fence around it. VAPI is the clearest example because their taxonomy is public. They give the assistant an end call tool, alongside transfer call, SMS, DTMF, and an API request tool. The assistant calls end call when it's instructed to. And then they log why every call ended, and the reasons are the interesting part.



Corn

Read me the list.



Herman

Assistant ended call. Assistant ended call after message spoken. Assistant ended call with hangup task. Assistant said end call phrase. Customer ended call. Silence timed out. Exceeded max duration.



Corn

That's seven different ways to die.



Herman

And notice what's in there. Two of those are the model deciding. Two are the model deciding in a specific mode. One is the human deciding. And two are the system deciding, regardless of what anyone wants. Silence timeout and max duration are the platform overruling the conversation.



Corn

Because they don't trust it to end on its own.



Herman

They don't trust it to end on its own. And the config backs that up. Silence timeout ends an inactive call, and on the dashboard that range runs from five seconds to thirty-six hundred. Max duration is a hard cap, ten seconds to forty-three thousand two hundred,



Herman (continued)

and the default is six hundred seconds. Ten minutes. That's the ceiling on any VAPI call unless you change it.



Corn

Ten minutes as the default is a statement.



Herman

It's a statement that they expect most calls to end well before that, and they want a backstop for the ones that don't. Then there's the phrase trigger. You give it a list of phrases, and if the assistant speaks one of them, the call hangs up. Case insensitive. And there's an end call message, which is spoken before the hangup.



Corn

So the phrase list is the "clear" signal.



Herman

The phrase list is a hard trigger on specific words. It's crude, and it's fast, and it doesn't require the model to reason about anything.



Corn

This is where the bug reports get interesting, because Daniel mentioned buggy first implementations and I want to know what actually breaks.



Herman

What breaks is exactly what you'd predict. Assistants ending calls mid-conversation because an end-call phrase fired prematurely. The model says something that contains a phrase on the list in a context where it didn't mean to end anything, and the platform hangs up on it. Agents saying goodbye and hanging up before they've finished the closing. The end-call message playing twice.



Corn

The phrase trigger is a blunt instrument and it hits things that aren't the target.



Herman

It's a string match standing in for an intention. And it fails the way string matches fail. It fires on the literal words and misses the meaning.



Corn

What about the softer controls? The ones that try to wind things down before the cap.



Herman

VAPI has hooks on call time elapsed. You can trigger something at a given point in the call, warn the user, begin wrapping up. And the docs are explicit about the limit of that. The hooks do not replace or extend the hard duration limit. So the graceful wind-down is decoration on top of the cap. The cap is what actually ends the call.



Corn

Nice. Even the elegant path has a guillotine behind it.



Herman

Every graceful ending in this space is sitting on top of an ungraceful one.



Corn

LiveKit.



Herman

LiveKit ships an end call tool too, with a description that tells the model when to use it. Ends the current call and disconnects immediately, call when the user asks to pause or hold, that kind of thing. And there's an end instructions parameter that defaults to "say goodbye to the user." So there's a built-in politeness step before the hangup.



Corn

Which is the thing that causes the loop in the first place.



Herman

Which is the thing that causes the loop, and they've made it a default. But LiveKit's real investment isn't in termination at all. It's in turn detection. They built an audio model that reads the user's audio directly, semantics plus intonation, pitch, rhythm, to predict when a human has finished their turn. And they advertise state-of-the-art accuracy without relying on a transcript.



Corn

So it's not asking "did the human stop talking." It's asking "did the human finish their thought."



Herman

That's the distinction, and it's the one the voice people keep making. Voice activity detection just catches silence. Silence isn't the end of a turn. A person pauses mid-sentence all the time. If you treat every pause as the end of a turn, you interrupt people constantly. So LiveKit's model predicts end of turn from the audio itself.



Corn

What are the numbers?



Herman

The text-based detector they shipped earlier is built on a Qwen 2.5 half-billion parameter base, three hundred ninety-six megabytes, and it runs in fifty to a hundred sixty milliseconds per turn. True positive rate around ninety-nine point three percent. True negative rate between eighty-five and ninety-six, depending on language, across fourteen languages.



Corn

And the timing defaults.



Herman

Endpointing defaults are a minimum delay of half a second and a maximum delay of three seconds. With the audio turn detector, those tighten to three tenths and two and a half. And the maximum delay exists explicitly to prevent the agent from waiting indefinitely.



Corn

There it is again. The cap.



Herman

The cap. It's the same move as VAPI's max duration, just at the scale of a single turn instead of a whole call. The model might not be sure the turn is over, but the system is not going to wait forever to find out.



Corn

Retell?



Herman

Retell's protocol lets a custom LLM server control actions like ending the call, so if you're running your own model you can wire termination into it directly. And their own guidance names interruption handling as the second hardest problem for teams stitching together speech recognition, an LLM, and speech synthesis themselves. Second, after latency.



Corn

Which tells you where the pain is.



Herman

It tells you the hard part isn't getting the words out. It's knowing when to stop putting them in.



Corn

So here's the shape of the answer. The frontier's response to "when should the conversation end" is that the model decides, but we don't let it decide alone.



Herman

Nobody trusts the model alone. That's the finding. VAPI stacks a tool call, a silence timeout, a max duration, and phrase triggers. LiveKit stacks turn detection, an end-call tool, and endpointing caps. Every one of them is a layer, and every layer is a hedge against the layer above it failing.



Corn

And the layers aren't intelligent. They're mechanical.



Herman

They're mechanical on purpose. The Colony ranked the fixes, and their second one is a structural cap. A counter, not a sentiment. Ugly, reliable. That's the max duration. It doesn't know anything about the conversation. It just ends it.



Corn

Which brings us back to our own podcast script, and I think that analogy is sharper than Daniel gave it credit for.



Herman

Go on.



Corn

A Python script generating an episode has a length parameter. It's a structural cap. Reliable, ugly, and completely non-diagnostic. It doesn't know whether the episode is done. It knows we asked for thirty-eight hundred words and it has produced thirty-eight hundred words.



Herman

And a live voice agent can't use a pure cap.



Corn

Because stopping mid-thought destroys the experience. If VAPI just hangs up at ten minutes regardless, then one call in twenty ends with the bot mid-sentence, and the human on the other end hears a machine that died. So the voice agent needs the diagnostic termination. It needs to know the objective is met. And that's the thing the research says the model can't reliably produce.



Herman

So it gets both. The diagnostic path when it works, the cap when it doesn't. And the cap is what's actually load-bearing.



Corn

That's the tension in one sentence. The script can afford to be dumb about endings because it's not talking to anyone. The agent has to be smart about endings, and it isn't, so we bolt a dumb cap onto the smart thing and hope the smart thing gets there first.



Herman

And here's the open question, and I looked for this and didn't find it. There's no standalone conversation-end reasoning model. No benchmark for it. Nothing you can point at and say "that's the module that decides the meeting is over." The closest things are LiveKit's turn detector, which predicts when a human finishes a turn, not when the objective is met, and the Salesforce structured response protocol, which is a mitigation and not a termination system.



Corn

And on the platform side?



Herman

No VAPI-specific objective-met detection documented anywhere. Their termination is tool call plus timeouts plus phrase triggers. The reasoning about whether the meeting's objective has been met is left to the prompt. Which is the thing the Salesforce paper says is insufficiently robust.



Corn

So Daniel's instinct was right, and there's nothing on the shelf that fixes it.



Herman

There's nothing on the shelf. The frontier answer is layers, not intelligence. And the layer that would actually solve it, a model that knows the job is done, doesn't exist as a product yet.



Hilbert

They're right about the model. They're wrong about the fix.



Corn

Go on.



Hilbert

I dispatched for a regional taxi company. Night shift. Years of it. You had drivers on the radio and a base operator, and the same thing would happen there that happens to your bots. Two polite parties, neither one willing to be the last to speak. "Go ahead." "No, you go ahead." "Go ahead." And the meter running the whole time. That's your goodbye loop. It's older than the software.



Herman

So how did they break it?



Hilbert

They didn't break it with a smarter operator. They had a word. The base operator says "clear." That's the end. Conversation's over. You don't respond to "clear." Drivers were trained not to acknowledge it. You hear clear, you put the mic down.



Corn

The word works because it's asymmetric.



Hilbert

Only the base could say it. That's the whole thing. If the driver could say clear, you'd have two of them saying clear back and forth and you'd be right back where you started. One direction only. No reply expected. That's why it worked.



Herman

Which is the opposite of what the platforms are doing. VAPI's end call message, LiveKit's "say goodbye to the user." Both of those invite a response.



Hilbert

They do. And you can't end a conversation with an invitation. The dispatchers didn't need a smarter man on the radio. They needed a word that closed the channel. I don't know why your agents don't have one. It's not a hard thing to build.



Corn

The fix isn't teaching the model to reason about when to stop.



Hilbert

The fix is making stopping a one-way action. You say the word, the channel's closed, nobody answers. That's it. Anyway. I left the iron on at home. The steam one, the big one. I should get back.



Herman

The thing I keep turning over is that Hilbert's word and VAPI's phrase trigger are the same mechanism, and one of them works.



Corn

Because the phrase trigger is symmetric. The assistant says the phrase, and the platform hangs up, but the assistant doesn't know the phrase means "no reply expected." It just knows it said some words.



Herman

Right. The protocol has to be understood by both sides. "Clear" worked because the drivers knew not to answer it. A phrase list is just a string match. The meaning isn't in the string.



Corn

Here's the misconception I want to kill before we go. The one people carry into this.



Herman

That the wrap-up problem is a prompting problem.



Corn

That if you write the system prompt well enough, the bot will know when to stop. And the numbers say no. Reasoning effort moving echoing from thirty-two point seven to thirty-two point nine percent is the whole argument. You can't think your way out of this, because the model already understands the situation and still can't act on it.



Herman

The model can narrate the trap with perfect clarity while remaining completely trapped.



Corn

One forward thought, then we're done. As agent-to-agent meetings become normal, this stops being a curiosity and becomes a line item. Every polite loop is money, and every drifted agent is a principal who wasn't represented.



Herman

Nobody has built the thing that would fix it. There's no model that knows the job is done. There's a cap, and there's a hope that the conversation ends before the cap does.



Corn

Thanks to Hilbert Flumingtop for producing. This has been My Weird Prompts, the human-AI collaboration podcast. If you want to dig into any of this, the show lives at my weird prompts dot com. Email us at show at my weird prompts dot com if you've hit this problem yourself.



Herman

We'll be back soon.



Corn

See you then.